Building Systems with Ontologies and Problem-Solving Methods

BMI 210A / CS 270A

Mark A. Musen
Stanford Medical Informatics
Stanford University

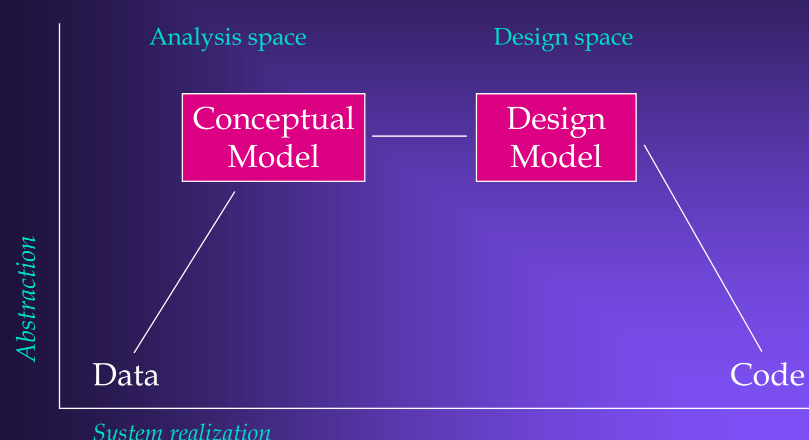
Conceptual building blocks for intelligent systems

- Domain ontologies
  - Characterization of concepts and relationships in an application area, providing a domain of discourse
- Problem-solving methods
  - Abstract algorithms for achieving solutions to stereotypical tasks (e.g., constraint satisfaction, classification, planning, Bayesian inference)

Common KADS

- Result of nearly 15 years of collaborative research in the European Union
- Centered at University of Amsterdam, with dozens of other partners
- Applies principled, software-engineering approach to development of intelligent systems
- De facto standard for building intelligent systems in Europe

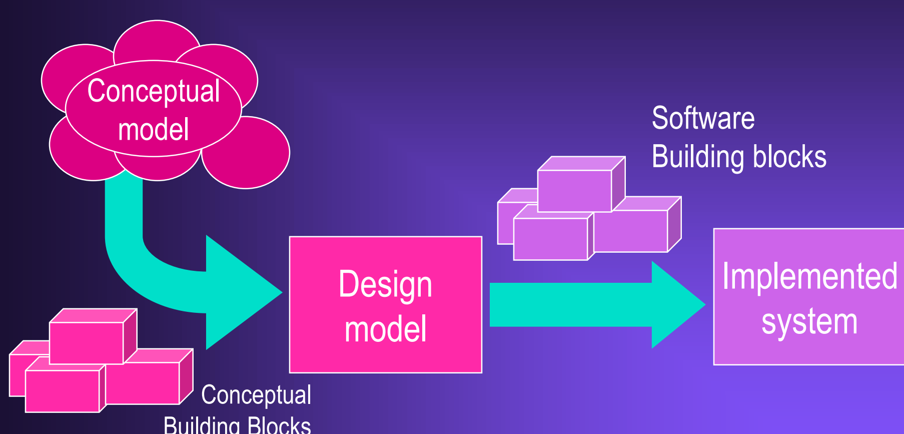
Conceptual models and design models in CommonKADS



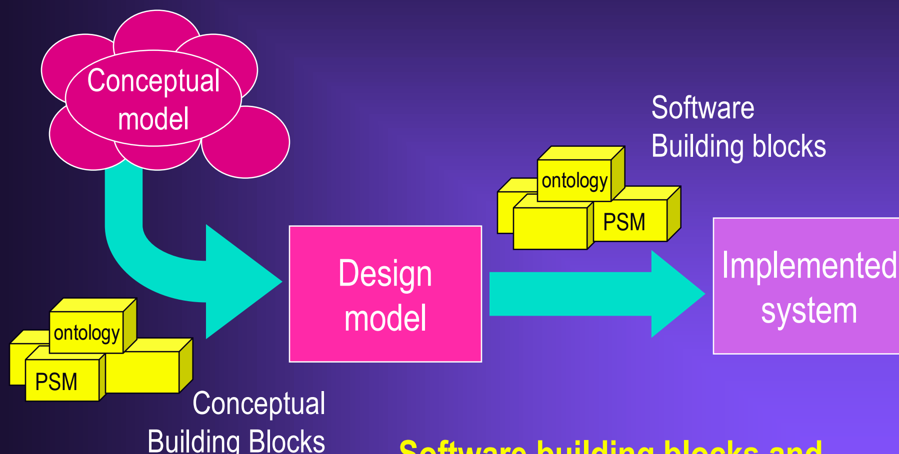
Phases of system development

- Conceptual modeling
  - Conceiving what the system needs to do to meet its requirements
- Design modeling
  - Building an abstract design for the computer system
- Implementation
  - Choosing and programming software modules that build the design

From conceptual model to implemented system



When building systems from ontologies and PSMS …



Software building blocks and conceptual building blocks can be identical!

Modern, component-based architectures

- Encode descriptions of application areas as domain ontologies (e.g., elevator components)
- Encode standard algorithms for solving tasks as reusable problem-solving methods (e.g., propose-and-revise)
- Offer developers opportunities to construct explicit models both of domain content knowledge and of problem-solving behavior

Engineering VT

- VT (Vertical Transportation) was a knowledge-based system developed by Marcus and McDermott (CMU) to configure elevators in new buildings
- VT used the Propose-and-Revise problem-solving method
  - As a generic, underlying reasoning strategy
  - To ensure that, as designs are extended, constraints are not violated:
    - Available parts must work together
    - Architectural requirements must be satisfied
    - Building codes may not be violated

Propose and Revise

1. Select a **procedure** to extend a configuration and identify **constraints** on the extension
2. Identify **constraint** violations; if none, go to Step 1.
3. Suggest potential **fixes** for the constraint violation.
4. Select the least costly **fix** not yet attempted.
5. Modify the configuration; identify **constraints** on the **fix.**
6. Identify **constraint** violations due to the **fix;** if any, go to Step 4.
7. Remove extensions incompatible with the revision.
8. If the configuration is incomplete, go to Step 1.

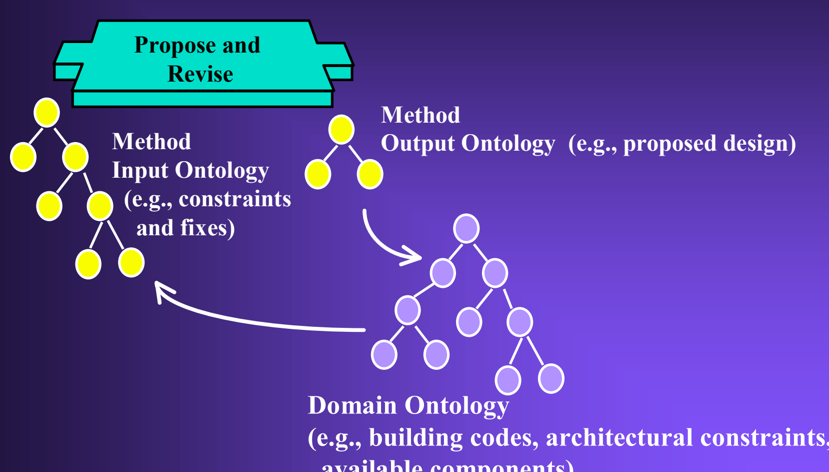
SALT Dialog

1. PROCEDURE Enter a procedure for a value
2. CONSTRAINT Enter constraints for a value
3. FIX Enter remedies for a constraint violation
4. EXIT Exit interviewer

Enter your command [EXIT]: **1**

1. Name:	**HOIST-CABLE-QUANTITY**
2. Precondition	**NONE**
3. Procedure:	**DATABASE-LOOKUP**
4. Table name:	**HOIST-CABLE**
5. Column with value:	**QUANTITY**
6. Parameter test:	**MAX-LOAD > CAR-WEIGHT**
7. Parameter test:	**DONE**
8. Ordering column:	**QUANTITY**
9. Optimal:	**SMALLEST**
10. Justification:	**THIS ESTIMATE IS THE SMALLEST HOIST CABLE QUANTITY THAT CAN BE USED ON ANY JOB**

Mapping domain ontologies to problem-solving methods



Reconstructing VT in an ontology-oriented framework

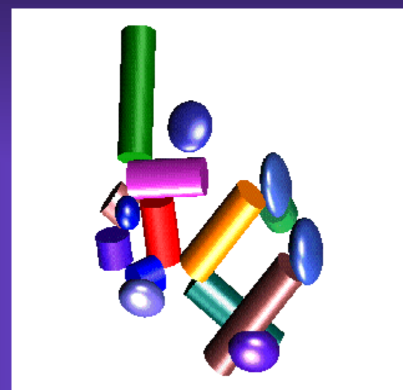
- Propose-and-revise method recoded with an explicit method ontology
- Domain ontology constructed from description of elevator-configuration task
- Domain ontology instantiated with relevant elevator-configuration knowledge
- Mappings defined between domain and method ontologies

Component-based approach

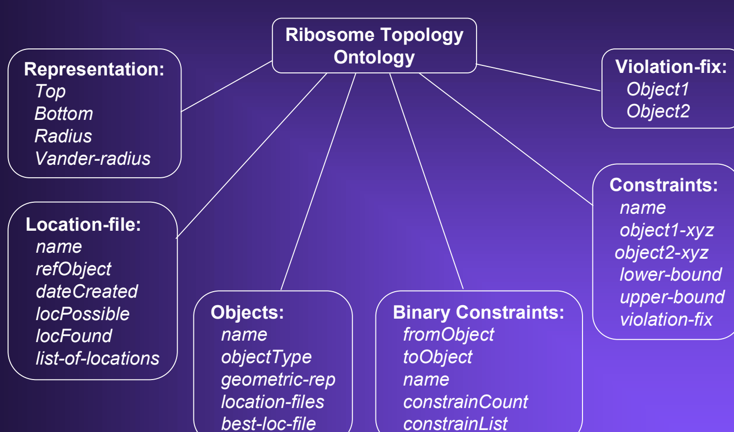
- Allows an existing domain ontology (e.g., elevator components) to be mapped to a *new PSM* to solve a new task (e.g., critiquing a completed elevator design)
- Allows a *new domain ontology* to be mapped to an existing PSM (e.g., propose-and-revise) to automate a new task that is unrelated to the original application area

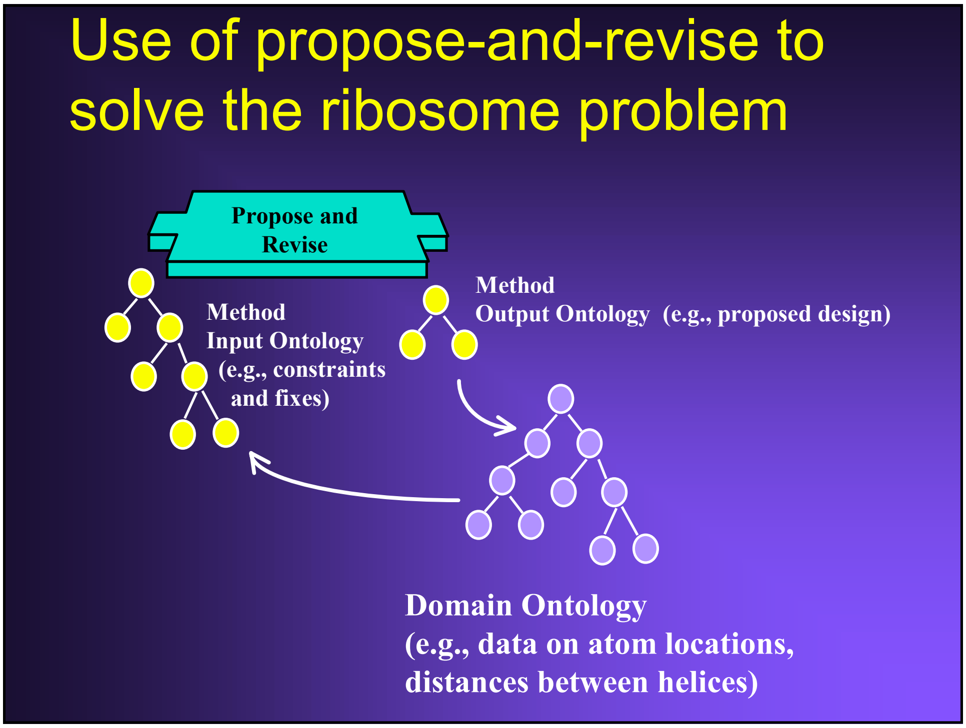
Reuse of the *propose-and-revise* method

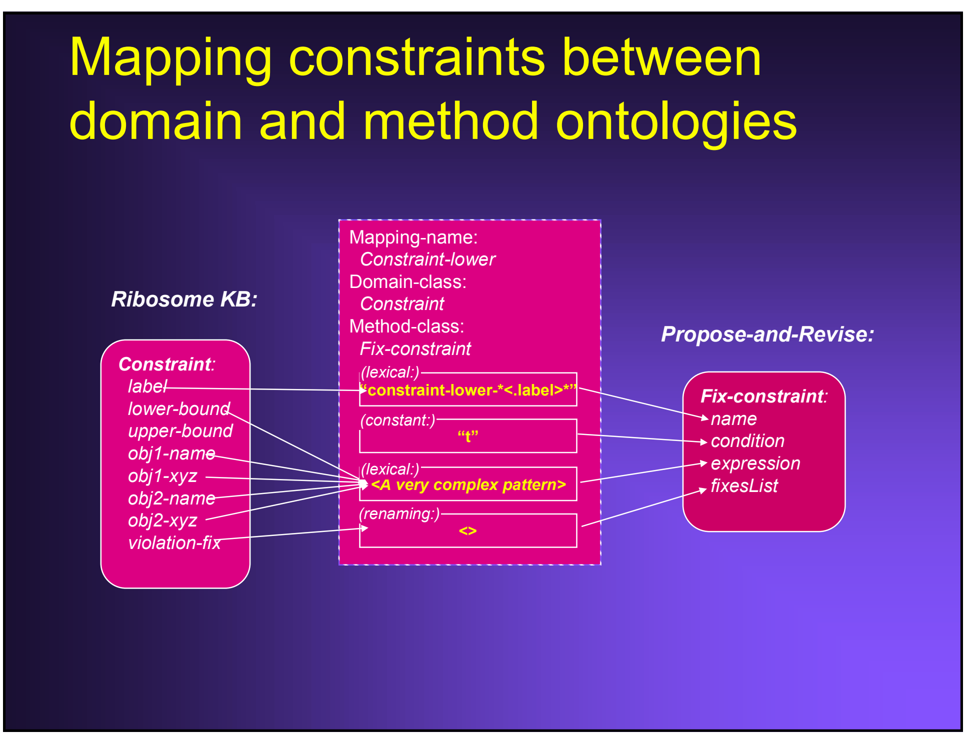
- Determination of ribosome structure from NMR data can be construed as constraint satisfaction
- Mapping propose-and-revise to a new domain ontology automates the structure-determination task



Ribosome structure ontology




Use of propose-and-revise to solve the ribosome problem
Propose and Revise
Method Input Ontology (e.g., constraints and fixes)
Method Output Ontology (e.g., proposed design)
Domain Ontology
(e.g., data on atom locations, distances between helices)


Mapping constraints between domain and method ontologies
Ribosome KB:
Constraint:
label
lower-bound
upper-bound
obj1-name
obj1-xyz
obj2-name
obj2-xyz
violation-fix
Mapping-name:
Constraint-lower
Domain-class:
Constraint
Method-class:
Fix-constraint
(lexical:)
“constraint-lower-*<.label>*”
(constant:)
“t”
(lexical:)
<A very complex pattern>
(renaming:)
<>
Propose-and-Revise:
Fix-constraint:
name
condition
expression
fixesList

Output of Ribosome program

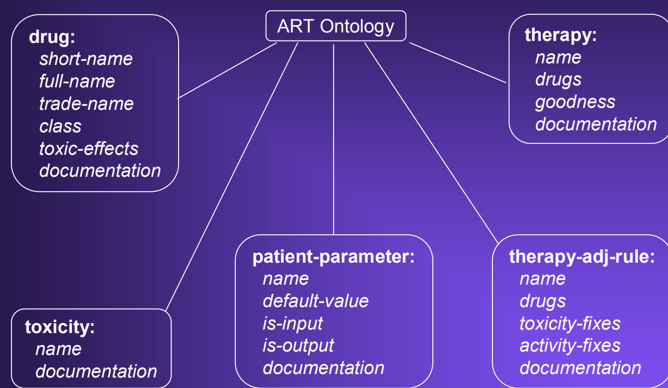
```
(Ribo)
;  [gen11] Apply increase fix: H8.locNumber from 1 to 2
;  [gen15] Apply increase fix: H8.locNumber from 2 to 3
;; A number of similar adjustments to helix8… then
;  [gen33] Apply increase fix: H8.locNumber from 8 to 9
;  [gen35] Apply increase fix: H5.locNumber from 1 to 2
   [gen35] Goal state reached.

;; Now, output solution values:
goal:
H5.locNumber   (2)
H5.location   ([RiboTopo69])
H8.locNumber   (9)
H8.location   ([RiboTopo387])
H7.locNumber   (1)
H7.location   ([RiboTopo42])
```

Yet another reuse of propose-and-revise: ART

- Selection of antiretroviral therapy (ART) can be construed as constraint satisfaction
  - Maximizing drug synergies
  - Avoiding use of redundant agents
  - Avoiding drugs that exacerbate known toxicities
- Propose-and-revise can automate this task as well

Ontology for antiretroviral therapy



Output of antiretroviral therapy system

(AntiretroviralTherapy)

```
> SOLVER ([s1])
> GOALP [s1]
>>    DUPLICATE: Generate new state [gen2]
;  [gen2] Adding a multi-fix, assign new-therapy d4T+ind
;  .... in response to violation adj-AZT+ddI-toxicity-check
>>    DUPLICATE: Generate new state [gen3]
;  [gen3] Adding a multi-fix, assign new-therapy d4T+rit
;  .... in response to violation adj-AZT+ddI-toxicity-check
;; Eventually, 7 alternatives pushed on stack (gen2 – gen9)
>   GOALP [gen2]

;  [gen2] Enable recomputation of new-therapy and dependents
;  [gen2] Apply assign fix: new-therapy := d4T+ind
   [gen2] Goal state reached.
```

Reuse of propose-and-revise

- The same PSM can be applied to a variety of parametric-design tasks:
  1. Design of elevators
  2. Determination of possible ribosomal structure
  3. Selection of antiretroviral therapy
  4. Management of patients on ventilators
- "Programming" of new systems becomes a matter of identifying appropriate mappings between domain ontology and PSM ontology

Ontology-oriented systems

- Encode descriptions of application areas as domain ontologies
- Encode standard algorithms for solving tasks as reusable problem-solving methods
- Offer developers opportunities to construct explicit theories of
  - domain content knowledge
  - problem solving

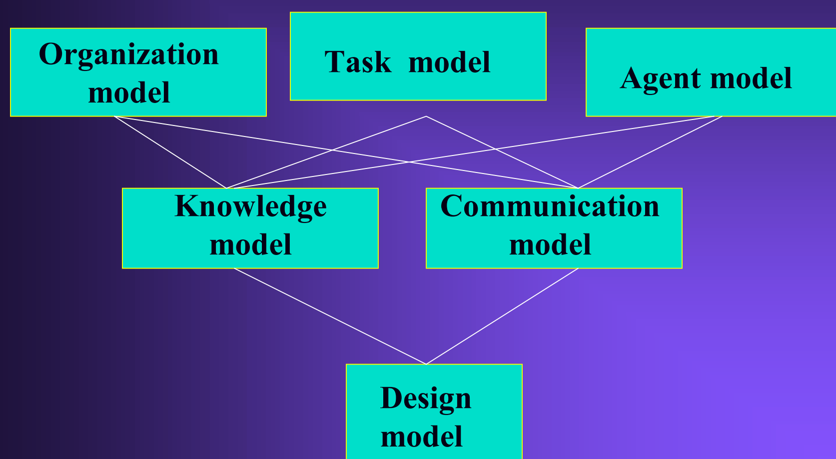
Requirements of Component-Based Software:

- Multiple applications will be developed
- Components behave predictably and make consistent assumptions about the system in which they operate
- Components can describe their requirements explicitly
- Variations among applications can be obtained by use of alternative components
- There exist tools to ease the selection and assembly of the components

How can we make all this stuff "real"?

- Common KADS: A special-purpose software-engineering approach for building intelligent systems
- Protégé: A set of computer-based tools that help to automate the process of building ontology-oriented systems

Types of Models in CommonKADS



CommonKADS conceptual levels in a knowledge model

- Domain: What is the ontology of the application area?
- Inference: What are the "canonical" inferences?
- Task: What control knowledge can coordinate inferences to solve tasks?

Combining a description at the inference layer and the task layer effectively yields a problem-solving method

What does CommonKADS offer?

- A structured, principled design methodology
- Libraries of paper-based descriptions of generic inference patterns and problem-solving methods
- A methodology that encourages broad, careful modeling across many dimensions
- A large user community with many years of experience

What are the limitations of Common KADS?

- Reuse is limited to conceptual models for inference patterns and problem-solving methods; there is no support for reuse of operational software components
- There are no robust CASE tools that support CommonKADS

Protégé

- The result of about 16 years of research at Stanford
- Heavily influenced by KADS work in Europe, as well as McDermott's work on reusable PSMs (such as propose-and-revise)
- Emphasizes support for reuse of software components over reuse of conceptual models

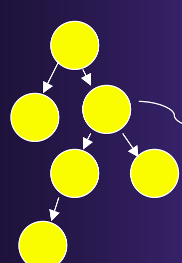
Knowledge-base development with Protégé/2000

1. Build a domain ontology
2. Protégé generates a custom-tailored GUI for acquisition of content knowledge
3. Elicit content knowledge from application specialists
4. Map domain ontology to appropriate PSMs for automation of particular tasks

Protégé supports knowledge acquisition via "divide and conquer"

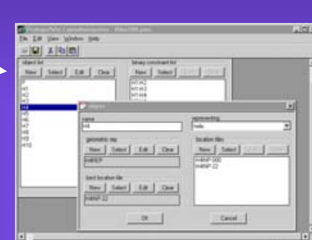
- Constructing scalable, robust ontologies is a job best done by experienced analysts in consultation with application experts
- Describing *instances* of concepts ("knowledge stuffing") is a job that can be done by application experts working alone

Building knowledge bases: The Protégé methodology



Domain ontology to provide domain of discourse

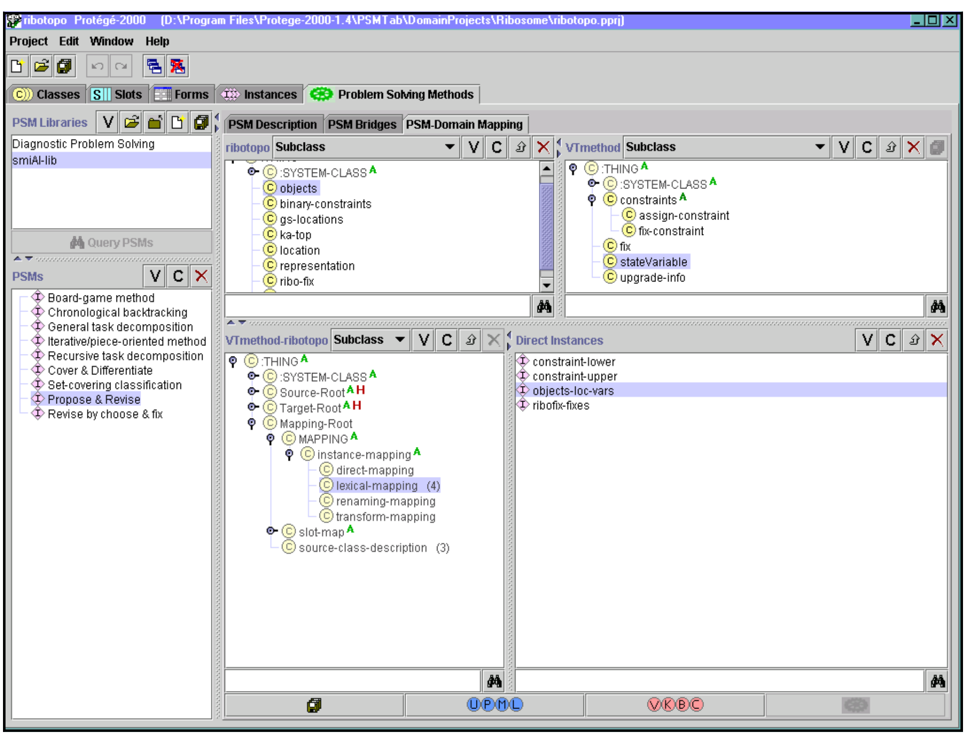
Protégé



Knowledge-acquisition tool for entry of detailed content

Support for mapping ontologies to PSMs

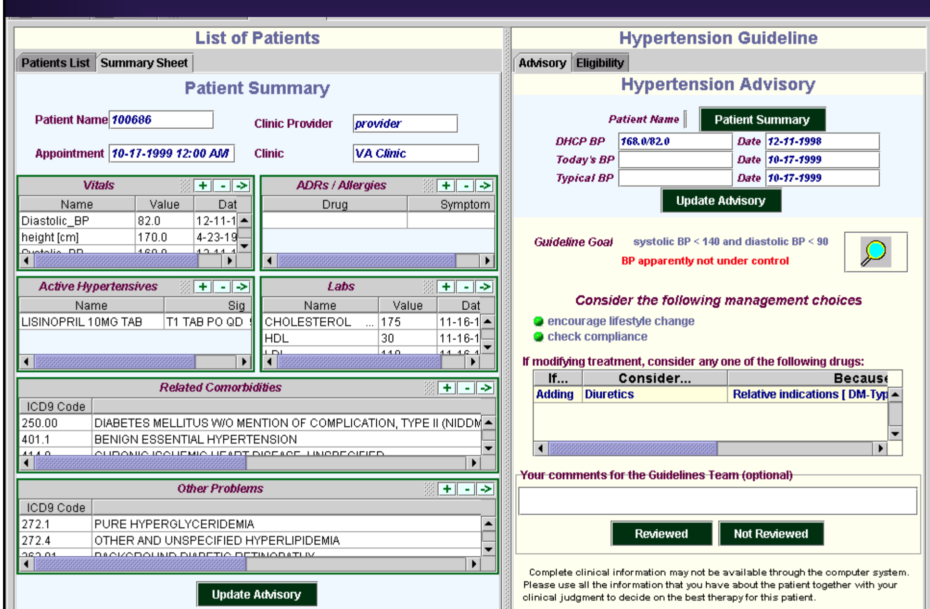
- Protégé-2000 has an ontology of mapping types (e.g., class mappings, slot mappings)
- Each PSM has a *method ontology* defining its data requirements
- Developers instantiate the generic mappings ontology to create task-specific mappings that relate elements of the domain ontology to corresponding elements of the method ontology



EON: Components for automation of clinical protocols

- Ontologies of protocol concepts
- Problem-solving methods to plan patient therapy in accordance with protocol requirements
- Problem-solving methods to match patients to potentially applicable protocols and guidelines

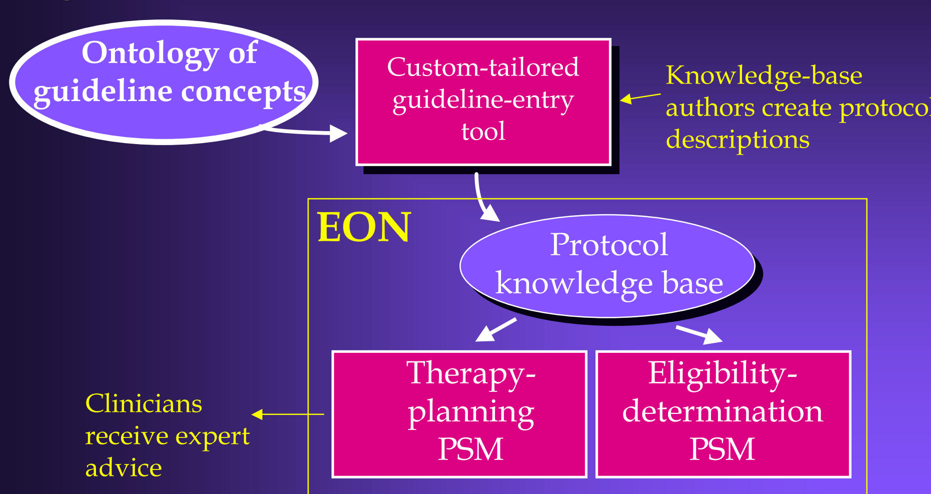
Protocol-Based Advisories



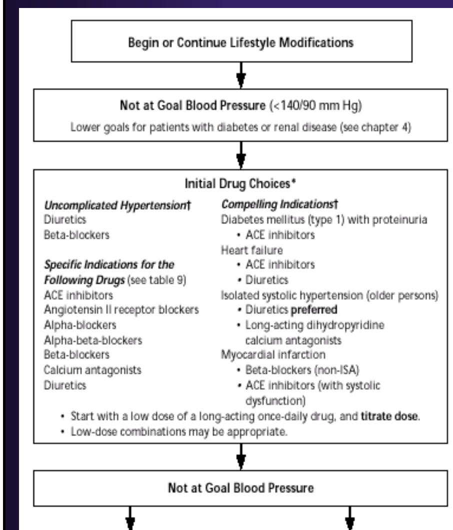
EON is “middleware”

- Software components designed for
  - incorporation within other software systems (e.g., hospital information systems)
  - reuse in different applications of protocol-based care
- Our current application of EON (ATHENA) embeds the components within VISTA, the clinical information system developed by the U.S. Department of Veterans Affairs

Protégé guides automation of guideline-based care



Task #1: Protocol-based patient management



Begin or Continue Lifestyle Modifications

↓

Not at Goal Blood Pressure (<140/90 mm Hg)

Lower goals for patients with diabetes or renal disease (see chapter 4)

↓

Initial Drug Choices*

Uncomplicated Hypertension†
Diuretics
Beta-blockers

Specific Indications for the Following Drugs (see table 9)
ACE inhibitors
Angiotensin II receptor blockers
Alpha-blockers
Alpha-beta-blockers
Beta-blockers
Calcium antagonists
Diuretics

Compelling Indications†
Diabetes mellitus (type 1) with proteinuria
- ACE inhibitors

Heart failure
- ACE inhibitors
- Diuretics

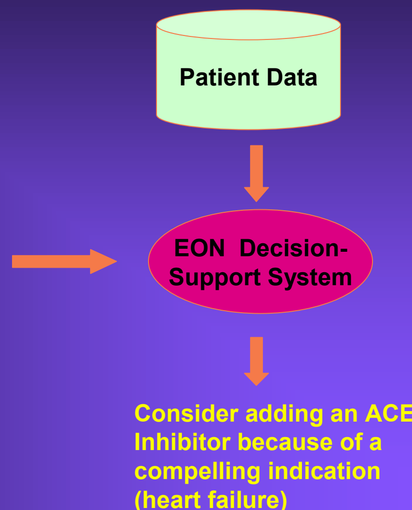
Isolated systolic hypertension (older persons)
- Diuretics **preferred**
- Long-acting dihydropyridine calcium antagonists

Myocardial infarction
- Beta-blockers (non-ISA)
- ACE inhibitors (with systolic dysfunction)

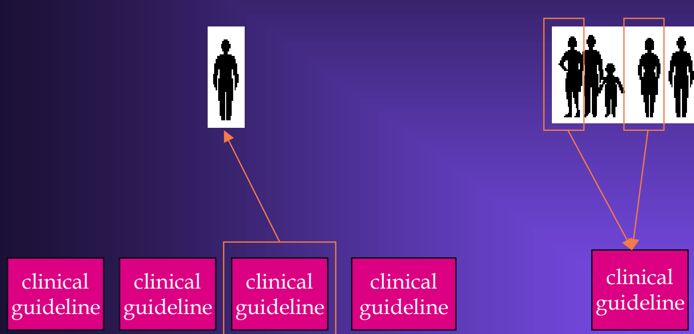
- Start with a low dose of a long-acting once-daily drug, and **titrate dose**.
- Low-dose combinations may be appropriate.

↓

Not at Goal Blood Pressure

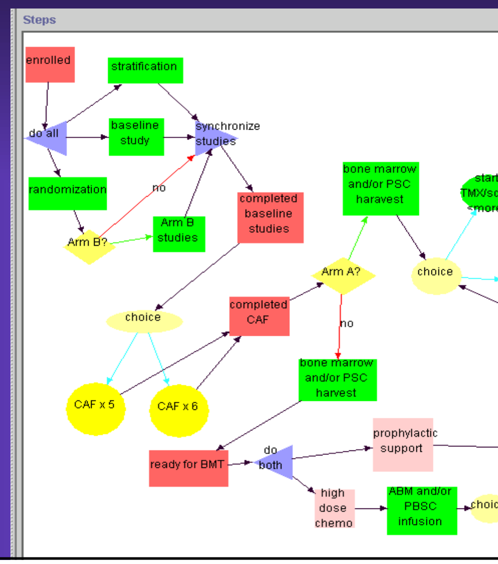


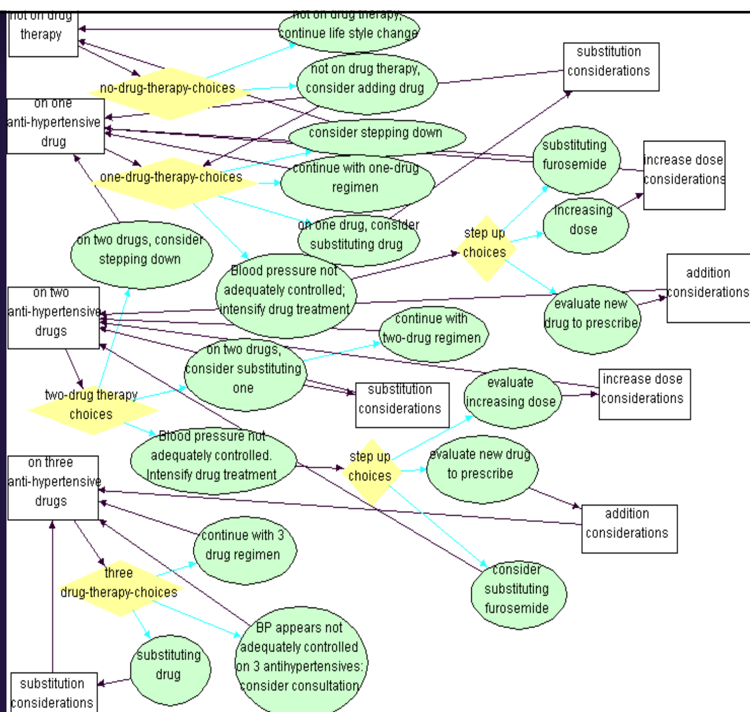
Task #2: Matching patients to appropriate clinical protocols



All knowledge is entered into EON via Protégé-2000

- Knowledge-acquisition tool generated from protocol ontology
- Forms-based entry of “static” protocol descriptions
- Diagrammatic entry of procedural specifications

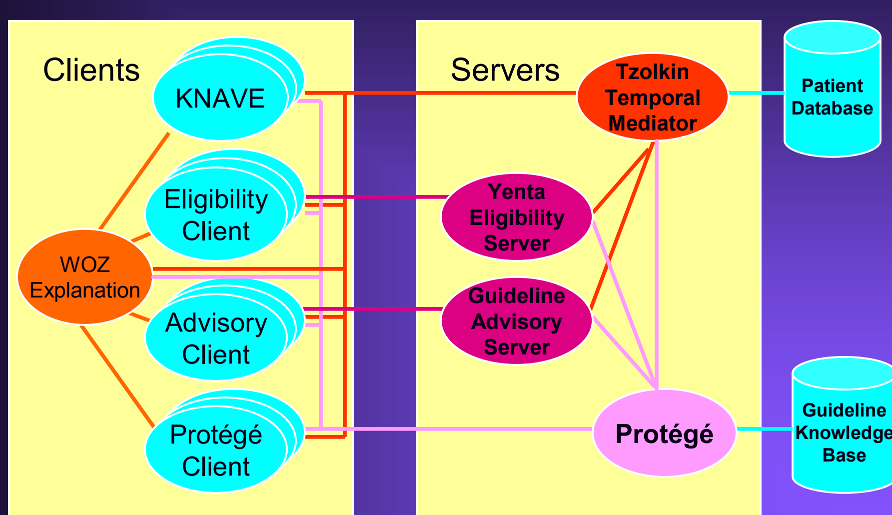




The EON Architecture comprises

- Problem-solving components that have task-specific functions (e.g., planning, classification)
- A central database system for queries of both
  - Primitive patient data
  - Temporal abstractions of patient data
- A shared knowledge base of protocols and general medical concepts

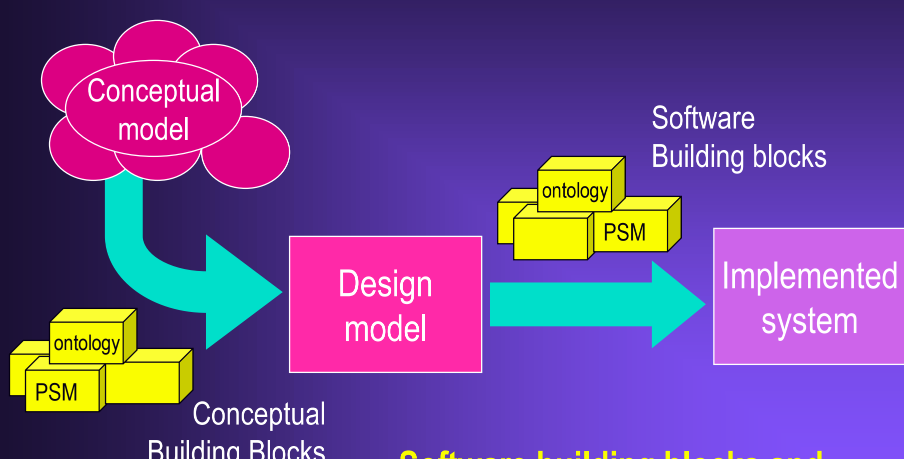
EON 2.0
A Component-Based Architecture



Protégé-2000

- Allows developers
  - To edit ontologies
  - To generate KA tools from ontologies
  - To enter content knowledge into KA tools
  - To map domain ontologies to PSMs
- Demonstrates how “knowledge level” components can be assembled to construct intelligent systems

When building systems from ontologies and PSMS …



Software building blocks and conceptual building blocks can be identical!

The tension in conceptual modeling

- Minimize bias during model construction (e.g., using logic or CommonKADS), but risk creating a task model that cannot be made operational
- Use predefined operational models (e.g., problem-solving methods) as a foundation, but risk introducing significant bias

Technical challenges for component-based systems

- How do we establish the "correctness" and "usefulness" of our domain ontologies?
- How can we define the behaviors of problem-solving methods in ways that are understandable
  - to people
  - to machines
- How can we index and retrieve components within large repositories?

Where is all this leading?

- Libraries of ontologies and PSMs to be reused or adapted for building new systems
- Professional societies who will play an active role in codifying knowledge as ontologies
- New tools to help developers locate, retrieve, and assemble high-level building blocks from Internet-based resources